საქართველოს ტექნიკური უნივერსიტეტი

ხელნაწერის უფლებით

ნინო კივილაძე

კორპორაციული Web-აპლიკაციების აგების ტექნოლოგიები სერვის-ორიენტირებული არქიტექტურით

წარდგენილია დოქტორის აკადემიური ხარისხის მოსაპოვებლად

წარდგენილი დისერტაციის

ა ვ ტ ო რ ე ფ ე რ ა ტ ი

სადოქტორო პროგრამა „ინფორმატიკა“ შიფრი 0401

თბილისი 2016 წელი

სამუშაო შესრულებულია საქართველოს ტექნიკური უნივერსიტეტის

ინფორმატიკისა და მართვის სისტემების ფაკულტეტის

„მართვის ავტომატიზებული სისტემების (პროგრამული ინჟინერია)“ დეპარტამენტში

სამეცნიერო ხელმძღვანელი:

პროფ. გ. სურგულაძე

რეცენზენტები:

პროფ. გიორგი ძიძიგური

ასოც.პროფ. ირაკლი ბულია

დაცვა შედგება 2016 წლის ” 2 ” ივლისი , 14.00 საათზე

საქართველოს ტექნიკური უნივერსიტეტის - „ინფორმატიკისა და მართვის სისტემების“ ფაკულტეტის სადისერტაციო საბჭოს კოლეგიის სხდომაზე, კორპუსი მე-4 , აუდიტორია 401

მისამართი: 0175, თბილისი, კოსტავას 77.

დისერტაციის გაცნობა შეიძლება სტუ-ს ბიბლიოთეკაში,

ხოლო ავტორეფერატისა - სტუ-ს ვებ-გვერდზე

სადისერტაციო საბჭოს

მდივანი: სრული პროფ. თინათინ კაიშაური

ნაშრომის ზოგადი დახასიათება

თემის აქტუალურობა. ადამიანური რესურსების მართვის სისტემა წარმოადგენს პროგრამულ უზრუნველყოფას, რომელიც მართავს ადამიანურ კაპიტალთან დაკავშირებულ ბიზნეს-პროცესებს. იგი წარმოადგენს ერთიან, კომპლექსურ სისტემას, რომელსაც იყენებს HR მენეჯერი გადაწვეტილებების მიღებისას. HR-ის ძირითადი ნაწილი მოიცავს ორგანიზაციებში მომუშავე თანამშრომლების შესახებ საბაზისო ინფორმაციას. იგი აერთიანებს პერსონალურ მონაცემებს, როგორიცაა თანამშრომლის მისამართი, დაბადების თარიღი, ოჯახური მდგომარეობა, საკონტაქტო ინფორმაცია; სახელფასო უწყისის მონაცემები; დანამატები, პრემია, ჯილდო, წახალისება; სამუშაოს აღწერილობა; ორგანიზაციული სტრუქტურა; თანამშრომლის პორტალი და self-service მოდული და ა.შ.

ადამიანური რესურსების მართვის ელექტრონული სისტემის გამოყენების ძირითადი მიზნებია:

1. ინფორმაციის გაუმჯობესებული მართვა. HR ადმინისტრატორს შეუძლია როგორც თანამშრომლების ინფორმაციის, ასევე დანამატებისა და თანამდებობრივი სარგოს მართვა.
2. თანამშრომლის პორტალი. HRMS სისტემა თანამშრომელს საშუალებას აძლევს განაახლოს მისი პირადი მონაცემები HR სპეციალისტის ჩარევის გარეშე. ეს HR სპეციალისტებს უთავისუფლებს დროს უფრო სტრატეგიული პროცესების დასაგეგმად.
3. მონაცემთა ცენტრალიზებული საათავსო. ინფორმაცია ერთ საცავში არის თავმოყრილი, რაც ზრდის ანგარიშგებების ეფექტურობას. გარდა ამისა, დოკუმენტების ერთიან სათავსოში ინახება თანამშრომლის ფაილები: სახელმძღვანელოები, ცნობები, დოკუმენტები, საბუთები და სხვა.
4. ნაკლებ სისტემებთან მუშაობა. HRMS სისტემაში ჩაშენებულია სახელფასო მოდული (Payroll), დოკუმენტბრუნვა (eDoc), საბუღალტრო სისტემა, ორგანიზაციის შიგა ინტრანეტი და სხვა პროგრამული დანართები.
5. თანამშრომლის განვითარება. HRMS სისტემის გამოყენება შესაძლებელია თანამშრომელთა განვითარების პროგრამების დაგეგმვისა და კადრების პროფესიული ზრდის თვალყურის დევნების მიზნით.
6. ანგარიშგებების მოდული. სისტემა შედგება ანგარიშგებებისგან, რომელთა გენერაცია დროის ნებისმიერ მომენტში არის შესაძლებელი და

ხელმძღვანელობას საჭირო ინფორმაციის მიღება HR სპეციალისტის ჩარევის გარეშე შეუძლია.

7. ორგანიზაციის ეფექტურობის გაზომვა. HRMS სისტემაში შესაძლებელია ადამიანური რესურსების გადინებისა და შემოდინების პროცესებზე დაკვირვება (თანამშრომლის აყვანა, თანამშრომლი ტრანსფერი, გათავისუფლება და სხვა).
8. თანამშრომლების დაქირავების მართვა. HRMS სისტემა შედგება თანამშრომელთა დაქირავების მოდულისგან, რომელშიც შესაძლებელია აპლიკანტებისა და მათი რეზიუმეების შესახებ ინფორმაციის თავმოყრა შემდგომი გამოყენების მიზნით.

საერთო ჯამში, ყველა ეს კომპონენტი აუმჯობესებს ადამიანური რესურსების მართვის სტრატეგიის დაგეგმვას, და ორგანიზაციის სამომავლო ხედვის დასახვას.

ნახ. 1 ადამიანური რესურსების მართვის სისტემის კომპონენტები

გარდა ზემოთ ჩამოთვლილი უპირატესობებისა, მნიშვნელოვანი პროგრესი შეინიშნება უსაფრთხოების თვალსაზრისითაც. ნაბეჭდი დოკუმენტაციის შენახვის ნაცვლად, ინფორმაცია ინახება ელექტრონული სახით ერთიან მონაცემთა საცავში და მასზე წვდომა დაშვებულია მხოლოდ შესაბამისი უფლებებამოსილებებით აღჭურვილი პირებისთვის. ამრიგად, ინფორმაციის გავრცელების შესაძლო გზები შემცირებულია, რაც, თავის მხრივ, ამცირებს ინფორმაციის გაჟონვის შესაძლო რისკებს.

ადამიანური რესურსების მართვის პროგრამულ უზრუნველყოფის იმპლემენტირება გადაწყვეტილია ვებ-ტექნოლოგიების გამოყენებით. ეს განაპირობა მრავალი მომხმარებლის მიერ (როგორც დედაქალაქიდან, ასევე რეგიონებიდან) სისტემაში წვდომის აუცილებლობამ. მომხმარებელი პროგრამულ დანართთან მუშაობისთვის საჭიროებს ვებ-ბრაუზერს და ინტერნეტ კავშირს. ეს მიდგომა უზრუნველყოფს ე.წ. „თხელი კლიენტების“ (კომპიუტერები, შეზღუდული აპარატურული შესაძლებლობებით) წვდომას ცენტრალიზირებული ინფრასტრუქტურის მქონე კომპლექსურ პროგრამულ უზრუნველყოფასთან. ამასთან ერთად, ვებ-ბრაუზერებისა და მათი მულტიმედიური შესაძლებლობების გამოყენებამ პროგრამისტებს გაუადვილა მდიდარი, ინტერაქტიული სამომხმარებლო ინტერფეისის შექმნა.

დღეს-დღეობით მკვეთრად მატულობს ბიზნეს-პროცესების კომპიუტერიზაცია და ელექტრონული სახით მართვა. ამასთან ერთად, პროგრამული უზრუნველყოფის იმპლემენტირებისას შეინიშნება ვებ-აპლიკაციების გამოყენების ტენდენცია. დესკტოპ-აპლიკაციებისგან განსხვავებით ვებ-აპლიკაციები მოსახერხებელია შემდეგი ფაქტორების გამო:

1. არ საჭიროებს გადმოწერას და ინსტალირებას. გაცილებით იოლია მომხმარებლის დაინტერესება პროგრამული უზრუნველყოფით, თუ მას მაშინათვე ექნება შესაძლებლობა ჩაერთოს სამუშაო პროცესში. დესკტოპ-აპლიკაციისგან განსხვავებით, ვებ-აპლიკაცია არ საჭიროებს კლიენტის მანქანაზე ჩასატარებელ წინასწარ სამუშაოებს, ინსტალაციას და კონფიგურირებას. მომხმარებელი პროგრამულ უზრუნველყოფასთან სამუშაოდ მხოლოდ ვებ ბრაუზერსა და ინტერნეტ-კავშირს საჭიროებს.
2. არ საჭიროებს განახლებებს. პროგრამული უზრუნველყოფის განახლება ხდება მხოლოდ ვებ-სერვერზე, ხოლო მომხმარებლები უკვავშირდებიან რა სერვერს, ავტომატურად მუშაობენ განახლებულ ვერსიში. ეს

გარანტიას იძლევა, რომ არც ერთი მომხმარებელი არ დარჩება ძველი ვერსიით და გამორიცხავს ვერსიებს შორის სხვაობით გამოწვეულ თავსებადობის პრობლემებს.

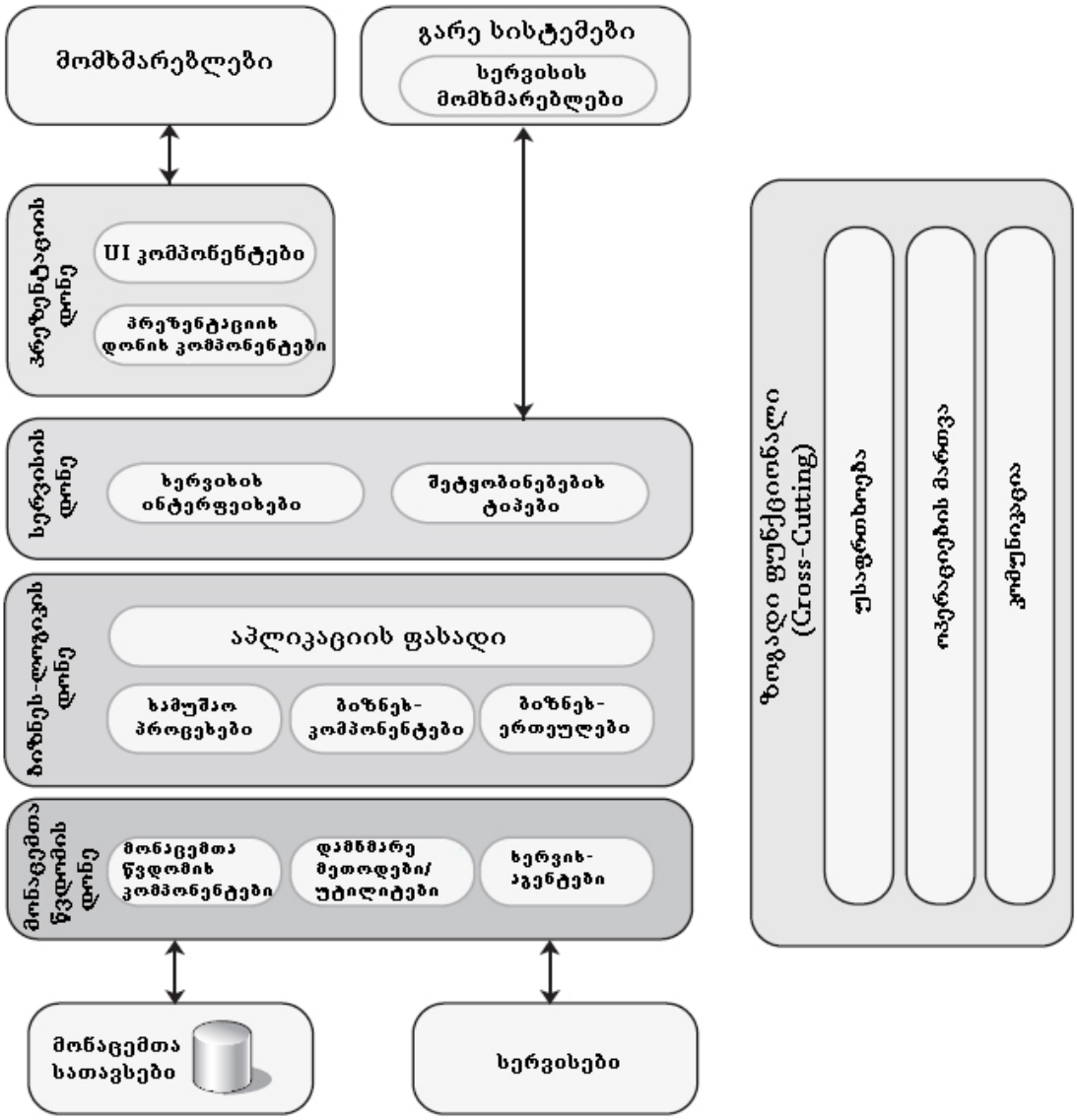
3. სიტემის გამოყენებადობის ანალიტიკა. ბიზნეს ხედვის კუთხით, დანახვა თუ პროგრამული უზრუნველყოფის რა მოდულებს იყენებენ მომხმარებლები დიდ უპირატესობას იძლევა. ამ სტატისტიკაზე დაკვირვებით შესაძლებელი გახდა მომხმარებლისთვის ნაკლებად საინტერესო პროგრამული მოდულების გამოვლენა, რომელთა გაუმჯობესებით კვლავ დავუბრუნებთ აქტუალობას ამა თუ იმ პროგრამულ კომპონენტს.
4. ნაკლები ხარჯი. პროგრამული უზრუნველყოფის შექმნის ხარჯი დესკტოპ-აპლიკაციების შემთხვევაში მატულობს, განსაკუთრებით, თუ მოთხოვნებში გათვალისწინებულია დანართის მუშაობა რამდენიმე პროგრამულ პლატფორმაზე (მაგ. OS X, PC და Linux).

მნიშვნელოვანია განვასხვაოთ ვებ-აპლიკაციები და ვებ-საიტები. ვებ-საიტი ინფორმაციული ხასიათისაა, იგი მომხმარებელს გარკვეული სახის ინფორმაციას აწვდის. ვებ-საიტი შეიძლება შევადაროთ საინფორმაციო ბრუშურას, რომელიც მომხმარებელს შეიძლება აწვდიდეს ურთიერთქმედების საშუალებას, მაგრამ შეზღუდული რაოდენობით. მეორეს მხრივ, ვებ-აპლიკაცია წარმოადგენს ინტერაქტიულ საიტს და შეიცავს ინტერაქტიულ ელემენტებს. ამის მაგალითებია Wikipedia, Facebook, Gmail და სხვა. ვებ აპლიკაციების დანიშნულებაა მომხმარებლებთან ურთიერთქმედება ინტერაქტიულ რეჟიმში. ვებ აპლიკაცია შედგება მდიდარი პროგრამული ლოგიკისგან და მომხმარებელს რაიმე დავალების ან ამოცანის შესრულების საშუალებას სთავაზობს.

სამუშაოს მიზანი და ამოცანები. დისერტაციის მიზანია მსხვილი ბიზნეს-პროცესების სამართავად სერვისზე ორიენტირებული ვებ-აპლიკაციების შემუშავების მეთოდოლოგიის გამოვლენა. განხილულია პრგორამული დანართის არქიტექტურის ძირიადი პრინციპები და მიდგომები. გამოვლენილია Silverlight ვებ-აპლიკაციების შემუშავებისას ჩასატარებელი სამუშაოები.

სტანდარტული პროგრამული დანართის არქიტექტურას ხშირად განმარტავენ, როგორც სისტემის ორგანიზების პროცესს, სადაც სისტემა წარმოადგენს სპეციფიური ფუნქციონალის შემცველი კომპონენტების კოლექციას. სხვაგვარად რომ ვთქვათ, არქიტექტურა ფოკუსირდება კომპონენტების

ორგანიზებაზე, რათა მათ იმპლემენტაცია გაუკეთონ სპეციფიურ ლოგიკურ ფუნქციონალს. ქვემოთ ილუსტრირებულია აპლიკაციის ზოგადი არქიტექტურა, სადაც კომპონენტები გადანაწილებულია სხვა და სხვა ლოგიკურ არეალში.



ნახ. 2 პროგრამული დანართის ზოგადი არქიტექტურა

პროგრამული დანართის ზოგადი სტრუქტურა მოიცავს ცალკეულ ლოგიკურ კომპონენტებში გადანაწილებულ კომპონენტებს. ლოგიკურ დონეს ევალება აპლიკაციის ფუნქციონალის ლოგიკური დაყოფა და არა კომპონენტების ფიზიკურ მისამართებზე დანაწილება. ლოგიკური შრეები შეიძლება განსხვავებულ ან საერთო ფიზიკურ დონეზე იმყოფებოდნენ. საკმაოდ ხშირია

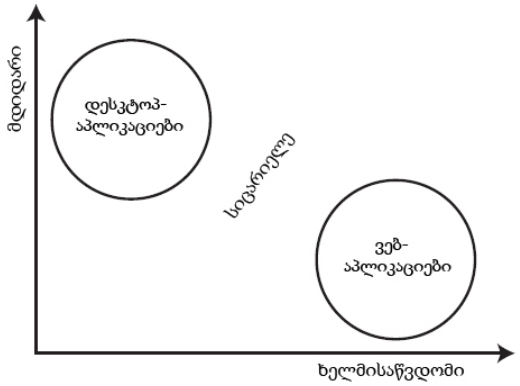
შემთხვევები, როდესაც რამდენიმე Layer-შრე იმყოფება ერთსა და იმავე ფიზიკურ მანქანაზე. გავრცელებული 3-დონიანი დიზიანი შედგება შემდეგი კომპონენტებისგან:

I. პრეზენტაციის დონე (Presentation Layer). ეს დონე შედგება მომხმარებელზე ორიენტირებული ფუნქციონალისგან, რომელიც პასუხისმგებელია მომხმარებლსა და სისტემას შორის ურთიერთქმედებაზე. პრეზენტაციის დონე ხიდის როლს თამაშობს და მომხმარებელს აწვდის ბიზნეს-ლოგიკის დონეზე აღწერილ ძირითად ფუნქციონალს.
II. ბიზნეს-ლოგიკის დონე (Business Layer). ამ დონეზე იმპლემენტირებულია სისტემის ძირითადი ფუნქციონალი და ინკაფსულირებულია შესაბამისი ლოგიკა. ზოგადად, ბიზნეს ლოგიკის დონე შედგება კომპონენტებისგან, რომლებიც იძლევიან სერვისის ინტერფეისებს, რათა სხვა გამომძახებელმა სისტემებმა შეძლონ მათი გამოყენება.
III. მონაცემთა წვდომის დონე (Data Layer). ამ დონეზე უზრუნველყოფილია სისტემის ფარგლებში დაჰოსტილ მონაცმთა სათავსოებზე და სხვა სისტემების მიერ მოწოდებულ მონაცემებზე წვდომა (შესაძლოა ეს წვდომა სერვისების გავლით ხდებოდეს). მონაცემთა წვდომის დონე იძლევა Generic ტიპის ინტერფეისებს, რომლებსაც მოიხმარენ ბიზნეს-ლოგიკის დონის კომპონენტები.

თანამედროვე ვებ-დაპროგრამების ტექნოლოგიებში დიდი პოპულარობა მოიპოვა Microsoft-ის კროს-პლატფორმულმა Silverlight ტექნოლოგიამ, რომელიც უზრუნველყოფს მდიდარ სამომხმარებლო ინტერფეისს (Rich Interactive Applications - RIA) და ინტერაქტიული აპლიკაციების შემუშავების საშუალებას იძლევა.

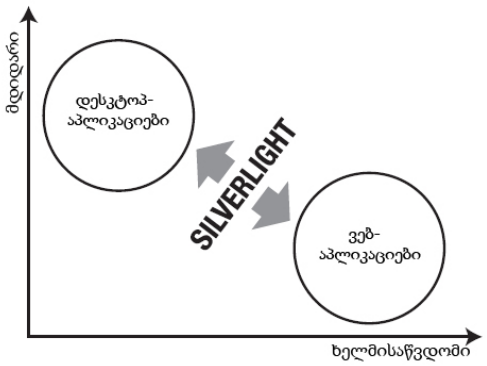
თანამედროვე ბაზარზე გაიზარდა მოთხოვნილებები აპლიკაციების მიმართ - პროგრამული უზრუნველყოფა უნდა აკმაყოფილებდეს არა მხოლოდ ფუნქციონალურ მოთხოვნილებებს, არამედ აგრეთვე უნდა ფლობდეს მდიდარ სამომხმარებლო ინტერფეისსაც. თუმცა მდიდარი სამომხმარებლო ინტერფეისი უშუალოდ პროგრამისტისთვის არ წარმოადგენს კრიტიკული მნიშვნელობის მქონე მახასიათებელს. ბოლო დროს აქტუალური ამოცანა გახდა ოქროს შუალედის პოვნა „მდიდარსა“ და „ხელმისაწვდომს“ შორის.

თუ განვიხილავთ სტანდარტულ დესკტოპ-აპლიკაციებს - აპლიკაციის ინსტალაცია ხდება ინდივიდუალურ კლიენტის მანქანაზე. დესკტოპ-აპლიკაცია იძლევა მდიდარ და ინტერაქტიულ სამომხმარებლო ინტერფეისს, მაგრამ აპლიკაციის ფუნქციონალური მახასიათებლები დამოკიდებული ხდება კლინტის მანქანაზე (სადაც იგი არის დაინსტალებული). აქედან თვალსაჩინოდ ჩანს, რომ დაბალია აპლიკაციის ხელმისაწვდომობის მახასიათებელი - ცალკეულ კომპიუტერზე უნდა ხდებოდეს ინსტალაცია და აპლიკაციიის სამომავლო მხარდაჭერა. თითოეულ პლატფორმისთვის აპლიკაციას უნდა გააჩნდეს ამ პლატფორმაზე მორგებული პროგრამული კოდი.



ნახ. 3 "მდიდარი" და "ხელმისაწვდომი" აპლიკაციების შედარება

ამის საპირისპიროა ვებ-აპლიკაციები, HTML-ზე ორიენტირებული პროგრამები, გათვლილი სხვადასხვა პლატფორმასა და ბრაუზერში სამუშაოდ. Microsoft პლატფორმაზე მომუშავე დეველოპერისთვის ეს ნიშნავს ASP.NET პროგრამული პაკეტის გამოყენებას და ვებ-სერვისების აგებას. აპლიკაციის ლოგიკის მხარის ძირითადი ნაწილი სრულდება სერვერზე, რაც აპლიკაციის სწრაფ მუშაობას უზრუნველყოფს. ამის საფასურია მწირი სამომხმარებლო გარემო - აპლიკაციას აქვს მაღალი ხარისხის ხელმისაწვდომობა, მაგრამ არც ისე „მდიდარი“ ინტერფეისი.



ნახ. 4 RIA ავსებს "მდიდარსა" და "ხელმისაწვდომ" აპლიკაციებს შორის სიცარიელეს

ტექნოლოგიების ამ ორ უკიდურესობას შორის თვალსაჩინო სიცარიელე არის წარმოქმნილი. ამ სიცარიელის ამოსავსებად დაპროგრამების ახალი მიდგომა იქნა შემუშავებული - RIA (Rich Internet Applications), რომელსაც გააჩნია ტრადიციული დესკტოპ-აპლიკაციების მსგავსი თვისებები და ფუნქციონალი.

RIA ტექნოლოგიების ერთ ერთი წარმომადგენელია Microsoft-ის Silverlight ტექნოლოგია.

როგორც ზემოთ განხილული შედარებიდან ჩანს, დასმული მიზანის მისაღწევად (ადამიანური რესურსების მართვის ელექტრონული სისტემა HRMS), გადაწყვეტილია მსხვილი ბიზნეს აპლიკაციების იმპლემენტირება ვებ-აპლიკაციის სახით, Silverlight ტექნოლოგიის გამოყენებით. მიზნის მისაღწევად აუცილებელია შემდეგი ძირითადი ამოცანების გადაწყვეტა:

- სერვისზე ორიენტირებული პროგრამული უზრუნველყოფის შემუშავებისთვის საჭირო დაპროგრამების პლატფორმის და ხელსაწყოების გამოვლენა.
- Silverlight აპლიკაციების შემუშავების დაგეგმვა და RIA აპლიკაციებთან საუშაო ეფექტური არქიტექტურული სტანდარტების გამოვლენა.
- პროგრამული უზრუნველყოფის ლოგიკურ კომპონენტებად დაყოფა და ერთმანეთისგან გამიჯვნა დამოუკიდებელ ლოგიკურ შრეებში.
- ლოგიკური კომპონენტების ერთმანეთთან დაკავშირება და კომუნიკაციის აწობა, ისე, რომ მათ შორის თანაკვეთის წერტილების მინიმალური რაოდენობა მივიღოთ. პროგრამული დანართის თითოეული კომპონენტი ან მოდული პასუხისმგებელი უნდა იყოს მხოლოდ ერთ კონკრეტულ ფუნქციონალზე.
- თეორიულად შემუშავებული არქიტექტურული მოდელების და მეთოდების პროგრამული რეალიზაცია Microsoft Visual Studio ხელსაწყოში, .NET Framework 5.0 პროგრამულ გარემოში, C# ენაზე, SQL Server, WCF, Silverlight, SQL Server Reporting Services პროგრამული პაკეტების გამოყენებით. ინტერაქტიული სამომხმარებლო გარემოს შემუშავება, რომელშიც ავტორიზებულ მომხმარებელს შეეძლება მუშაობა, ინფორმაციის შეტანა და ანგარიშგებების ამოღება.
- Silverlight ტექნოლოგიასთან სამუშაო გარემოს გამართვა და კომპანია Microsoft-ის მიერ შემოთავაზებული ინსტრუმენტების ინსტალაცია
    - Silverlight Tools for Visual Studio – პაკეტით Visual Studio ხელსაწყოს გამდიდრება Silverlight პროექტების მართვისთვის საჭირო ფუნქციონალით.
    - Expression Blend 4: გრაფიკული ხელსაწყო XAML-ზე აგებული სამომხმარებლო ინტერფეისების ასაწყობად. Expression Blend-ი არ არის სავალდებულო ხელსაწყო, მაგრამ Visual Studio-სთან შედარებით უფრო მდიდარი სამომხმარებლო ინტერფეისის შემუშავებას უზრუნველყოფს.
    - Silverlight Toolkit: უფასო პროექტი, მოიცავს დამატებით Silverlight კონტროლებს.

- View და ViewModel დონეების ურთიერთკავშირის გამართვა. პროგრამის უკანა მხარეს მონაცემთა ცვლილების შემთხვევაში View ინტერფეისზე მონაცემების ავტომატური განახლება (ან პირიქით: აპლიკაციის წინა მხარეს მონაცემების ცვლილების შემთხვევაში - უკანა, პროგრამული ლოგიკის მხარეს შესაბამისი ცვლადების ავტომატური განახლება).
- ვებ-აპლიკაციაში სერვისების ასინქრონული გამოძახების მოდელის იმპლემენტირება და ხანგრძლივი გამოთვლითი პროცესების შემთხვევაში ბრაუზერის ბლოკირების თავიდან არიდება. მომხმარებლის ინფორმირება გამოთვლით სამუშაოების მსვლელობის შესახებ (ეკრანზე რაიმე შეტყობინების ან ვიზუალიზაციის გამოტანა, მაგ. Busy Indicator კონტროლი).
- ბიზნეს წესების მართვა და ვალიდაციების მექანიზმების გამოყენება. ვალიდაციების იმპლემენტაცია როგორც კლიენტის, ასევე სერვერის მხარეს.
- უსაფრთხოების საკითხების გათვალისწინება, სახიფათო კოდებისა და ჰაკერული თავდასხმების არიდება, სენსიტიური ინფორმაციის დაცვა. პროგრამულ დანართში მომხმარებლის ქმედებების ლოგირების და აუდიტის ფუნქციონალის დაგეგმვა, განსაზღვრა, თუ რა ტიპის ინფორმაცია უნდა მოხვდეს ლოგირების ცხრილში.
- მომხმარებლის აუტენტიფიკაციის გამართვა. აუტენტიფიკაციის შესაძლო ვარიანტების გაანალიზება: მომხმარებლები ერთი და იმავე უფლებამოსილებით (Credentials) დაიშვებიან რამდენიმე პროგრამულ დანართში თუ არა, სერტიფიკატზე დაფუძნებული აუტენტიფიკაციის სისტემა გამოიყენება თუ არა და ა.შ.
- პროგრამული დანართის სერვერზე განთასება. ჩასატარებელი სამუშაოების დაგეგმვა, როდესაც ბრაუზერის Plug-In არ არის დაინსტალირებული. აპლიკაციის ხელახლა განთავსების შემთხვევის გათვალისწინება, მაშინ, როდესაც აპლიკაცია გაშვებულია კლიენტის მანქანაზე. აპლიკაციის ლოგიკურ მოდულებად დაყოფა და კომპონენტების ჩანაცვლება (მთლიანად აპლიკაციის გამოცვლის გარეშე). ცალკეულ კომპონენტებზე ვერსიების განსაზღვრა (Versioning).

კვლევის ობიექტი. ადამიანური რესურსების მართვის ბიზნეს-პროცესები. სერვისზე ორიენტირებული პროგრამული უზრუნველყოფის იმპლემენტაციის არქიტექტურული სტანდარტები და მეთოდები. ვებ-აპლიკაციების შემუშავების ზოგადი სტანდარტების გამოვლენა და პროგრამული უზრუნველყოფის დაპროექტება საუკეთესო პრაქტიკების გათვალისწინებით. სერვისზე ორიენტირებული არქიტექტურის რეალიზაცია WCF სერვისებისა და პროქსი კლასების საფუძველზე. Silverlight გარემოში ინტერაქტიული და ეფექტური დიზაინის შემუშავების მეთოდოლოგია. ვებ-აპლიკაციიდან ანგარიშგებების გენერაცია და ექსპორტირება სხვა და სხვა ფორმატით. სტატისტიკური

მონაცემების ანალიზი და ინფორმაციის გამოტანა ანგარიშგებებში. უსაფრთხოების მართვა, ავტორიზაცია, ლოგირება და შეცდომებზე რეარგირება. პროგრამული ხარვეზების დიაგნოსტიკა და აღმოფხვრა ავტომატიზირებული ტესტირების სისტემების საშუალებით.

მეცნიერული სიახლე. ნაშრომში წარმოდგენილია შემდეგი სიახლეები:

1. შემოთავაზებულია თანამედროვე ბიზნეს-პროცესების გადაწყვეტა სერვისზე ორიენტირებული ვებ-აპლიკაციების საფუძველზე, სადაც მდიდარ სამომხმარებლო გარემოს განაპირობებს Silverlight ტექნოლოგიის გამოყენება. პრაქტიკული რეალიზება წარმოდგენილია ადამიანური რესურსების მართვის ერთიანი ელექტრონული სისტემის (eHRMS) სახით. eHRMS პროგრამული უზრუნველყოფა სახელმწიფო ორგანიზაციებში წარმატებით სარგებლობს. იგი დანერგილია ორმოცზე მეტ ორგანიზაციასა და განყოფილებაში.

2. შემუშავებულია RIA აპლიკაციების დაპროექტების მეთოდოლოგია. წარმოფგენილია არქიტექტურული სტანდარტების კრებული, რომელთა გამოყენება განაპირობებს მდიდარი პროგრამული ლოგიკისა და ინტერაქტიული სამომხმარებლო გარემოს მუშაობას ვებ-გარემოში.

3. გამოვლენილია Silverlight-აპლიკაციებში ანგარიშგებების მოდულის ჩაშენების მეთოდები. ასევე დემონსტრირებულია Word დოკუმენტის შაბლონების გამოყენებით ფაილების გენერაცია და ექსპორტირება სხვა და სხვა ფორმატში.

4. შემუშავებულია უსაფრთხოების მართვისა და ლოგირების მოდელი. დემონსტრირებულია სერვისზე ორიენტირებული არქიტექტურის საშუალებით სხვა პროგრამულ დანართებთან კომუნიკაცია, ინფორმაციისა და ფუნქციონალის მიღების მიზნით: პიროვნების პირადი ნომრისა და გვარის საფუძველზე საჯარო რეესტრის სერვისიდან პირადი ინფორმაციის მიღება; უფლებების მართვის ელექტრონულ სიტემაში (ePassport) მომხმარებლების როლებისა და უფლებების დარეგისტრირება და HR სისტემაში მათი ინტეგრაცია.

შედეგების გამოყენების სფერო. დისერტაციის შედეგებს აქვს პრაქტიკული ღირებულება. ისინი შეიძლება გამოყენებულ იქნას რეალური ამოცანების გადასაჭრელად, კერძოდ კი, RIA ტექნოლოგიაზე Silverlight გარემოში მდიდარი სამომხმარებლო ინტერფეისის შემუშავების მიზნით. ასევე, პროგრამული დანართის შემუშავების დროს არქიტექტურული გადაწყვეტილებების მიღებისა და დაგეგმვის ეტაპზე.

ნაშრომის აპრობაცია: დისერტაციის ძირითადი შინაარსი მოხსენებული იყო ინფორმატიკისა და მართვის სისტემების ფაკულტეტის „მართვის ავტომატიზებული სისტემების (პროგრამული ინჟინერია)“ კოლეგიის სამეცნიერო სემინარების სხდომებზე, ასევე:

- VII საერთაშორისო სამეცნიერო-პრაქტიკულ კონფერენციაზე „ინტერნეტი და საზოგადოება“, INSO-2015, 11 ივლისი, აკ.წერეთლის სახ. უნივერსიტეტი, ქუთაისი. 2015;
- III საერთაშორისო სამეცნიერო კონფერენციაზე „კომპიუტინგი/ ინფორმატიკა, განათლების მეცნიერებები”. ბათუმი. 17-19.ოქტ., 2014;
- აკადემიკოს ი.ფრანგიშვილის დაბადების 85 წლისთავისადმი მიძღვნილი საერთაშორისო სამეცნიერო კონფერენციაზე „საინფორმაციო და კომპიუტერული ტექნოლოგიები, მოდელირება, მართვა“. თბილისი, 5 ნოემბერი, 2016 წელი.

პუბლიკაციები: დისერტაციის ძირითადი შედეგები გამოქვეყნებულია 5 სამეცნიერო ნაშრომში, რომელთა ჩამონათვალიც მოყვანილია დისერტაციის და ავტორეფერატის ბოლოს.

ნაშრომის მოცულობა და სტრუქტურა: დისერტაციის სრული მოცულობა შეადგენს 126 ნაბეჭდ გვერდს; შედგება რეზიუმეს (ორ ენაზე), სარჩევის, შესავლის, სამი თავის და დასკვნისგან. ახლავს 64 ნახაზი, 2 ცხრილი და 50 გამოყენებული ლიტერატურის სია.

დისერტაციის მოკლე შინაარსი

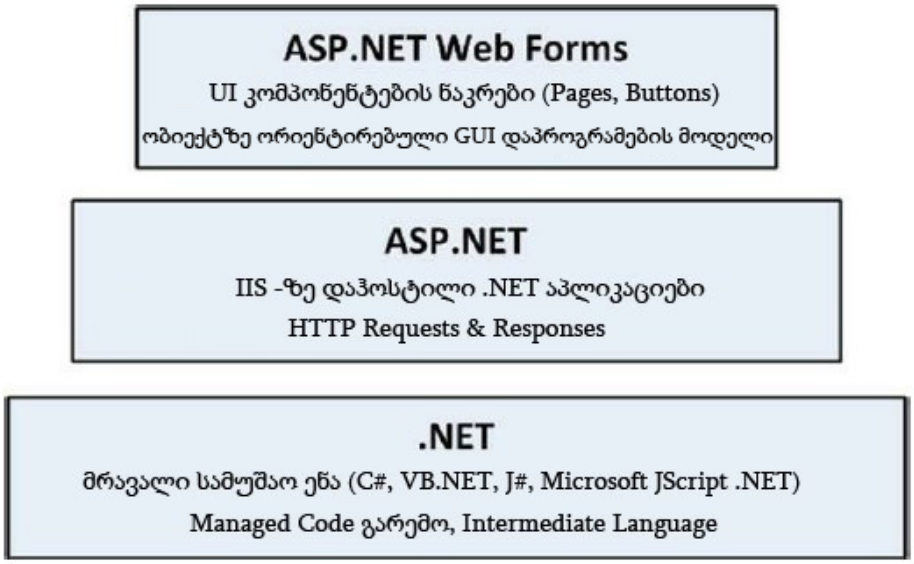
დისერტაციის **პირველი თავი** ეხება თანამედროვე ვებ-ტექნოლოგიების მიმოხილვას. გაანალიზებულია ASP.NET Web Forms, Silverlight, MVC ტექნოლოგიები, შედარებულია მათი ფუნქციონალური მახასიათებლები, გამოვლენილია დადებითი და უარყოფითი მხარეები.

განხილულია ASP.NET ვებ-ტექნოლოგიების განვითარების შკალა. 2002 წელს კომპანია Microsoft-მა შეიმუშავა .NET Framework-ის 1.0 ვერსია, რომელსაც მოყვა Active Server Pages (ASP) ტექნოლოგიის შემუშავება. ASP.NET აგებულია Common Language Runtime (CLR) გარემოზე, რაც პროგრამისტს ASP.NET კოდის ნებისმიერ .NET ენაზე (C#, Visual Basic .Net) დაწერის საშუალებას აძლევს. ASP.NET ტექნოლოგია მუშაობს HTTP პროტოკოლზე და HTTP ბრძანებებით ახდენს ორმაგ კომუნიკაციას ბრაუზერსა და სერვერს შორის. ASP.NET აპლიკაცია წარმოადგენს კომპილირებულ კოდს, რომელიც იყენებს .NET გარემოში არსებულ კლასებსა და იერარქიულ სტრუქტურებს.

ASP.NET Web Forms ტექნოლოგიის გამოშვებით კომპანია Microsoft-მა კიდევ უფრო განავრცო ვებ-აპლიკაციის ხდომილებებით მართული მოდელი. ბრაუზერი გადასცემს ვებ-ფორმის ინფორმაციას სერვერს და სერვერი საპასუხოდ აბრუნებს დამუშავებულ HTML გვერდს. HTTP პროტოკოლის მუშაობის დასამალად და HTML კონტენტის გენერაციის გასაიოლებლად

შეიმუშავეს სამომხმარებლო ინტერფეისის სპეციალური კომპონენტები - სერვერული მხარის კონტროლები. მათ უპირატესობად ითვლებოდა:

- HTTP მოთხოვნებს შორის მდგომარეობის ამსახველი ინფორმაციის შენახვა ViewState ობიექტში.
- HTML ფორმატში დარენდერება და კლიენტის მხარის ხდომილებების ავტომატურად დაკავშირება შესაბამის სერვერულ პროგრამულ ლოგიკასთან.



ნახ. 5 ASP.NET Web Forms ტექნოლოგიის განვითარების შკალა.

როგორც აღმოჩნდა, ტრადიციული ASP.NET Web Forms ტექნოლოგია დადებით მხარეებთან ერთად, უფრო გართულდა. ამის მიზეზები იყო: ViewState ობიექტის წონა, დიდი ზომის მონაცემთა ბლოკების ტრანსფერი კლიენტსა და სერვერს შორის; გვერდის სასიცოცხლო ციკლის სირთულე; მცდარი ილუზია განცალკევებული კონცეფციების შესახებ; HTML მარკირებაზე შეზღუდული კონტროლი; ტესტირების სისტემების დაბალი მხარდაჭერა.

Web Forms ტექნოლოგიის მიმართ აგორებული კრიტიკის საპასუხოდ კომპანია Microsoft-მა შეიმუშავა ახალი პროგრამული პლატფორმები. 2007 წლის ოქტომბერში ოფიციალურად გამოვიდა ASP.NET MVC ტექნოლოგია. მასში კომბინირებულია model-view-controller (MVC) არქიტექტურის ეფექტურობა, Agile Development მიდგომის უახლესი იდეოლოგია და ASP.NET პლატფორმის საუკეთესო ნაწილები. MVC ტექნოლოგიის უპირატესობებია: MVC არქიტექტურა, განვრცობადობა, HTML კოდზე და HTTP პროტოკოლზე მაღალი ხარისხის კონტროლის მექანიზმები, ტესტირებადობა, მარშრუტიზაციის მძლავრი სისტემა და ASP.NET პლატფორმის საუკეთესო ნაწილების გამოყენება.

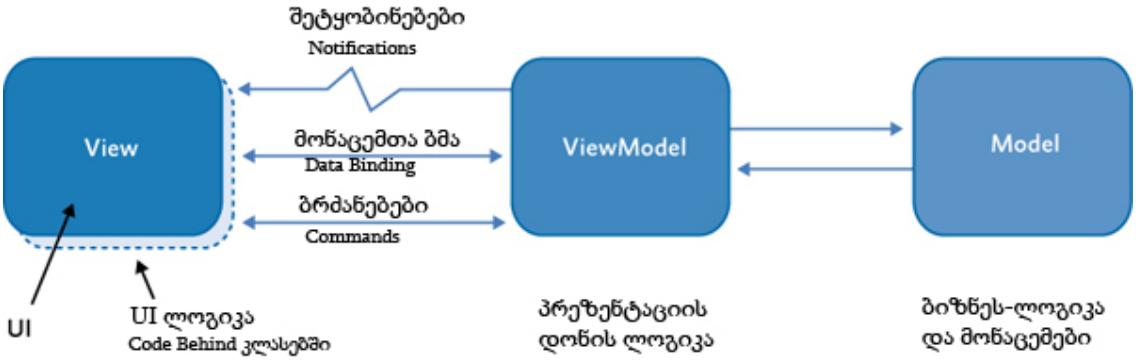
MVC ტექნოლოგიის შეზღუდვებად მოიაზრება მდიდარი სამომხმარებლო ინტერფეისის კონტროლების ნაკლებობა და ინტერაქტიულობის მისაღწევად ლოგიკის იმპლემენტირება თითქმის თავიდან. გარდა ამისა, სხვადასხვა კონტროლის მდგომარეობის შენახვა-შენარჩუნება Web-Forms პლიკაციებთან შედარებით რთული მისაღწევია ViewState ობიექტის არ არსებობის გამო.

ამ პრობლემებს წყვეტს კომპანია Microsoft-ის კიდევ ერთი ახალი ვებ-დაპროგრამების ტექნოლოგია Silverlight, რომელიც უზრუნველყოფს მდიდარ სამომხმარებლო ინტერფეისს (Rich Interactive Applications - RIA) და ინტერაქტიული აპლიკაციების შემუშავების საშუალებას იძლევა. იგი კარგად ესადაგება თანამედროვე პროგრამული უზრუნველყოფების მიმართ წაყენებულ მოთხოვნებს: განაპირობებს სამომხმარებლო ინტერფეისის სიმდიდრესა და ხელმისაწვდომობის მაღალ ხარისხს. RIA ტექნოლოგია შეიცავს Runtime პაკეტს, რომელიც ეშვება კლიენტის კომპიუტერზე და არქიტექტურულად ზის მომხმარებელსა და სერვერს შორის.

Microsoft-ის RIA ტექანოლოგიაში Silverlight-ი წარმოადგენს Client Runtimes-ს. Silverlight-ტექნოლოგია არის კროს-პლატფორმული, კროს-ბოუზერ დანართი (plug-in), რომელშიც სამომხმარებლო ინტერფეისი და გრაფიკული ელემენტები რენდერდება სპეციალურ არეზე/შრეზე (canvas), რაც შემდომში ჩაშენებადია HTML-გვერდში. Silverlight-აპლიკაციების XAML-ის შესამუშავებლად მოსახერხებელია Microsoft-ის გრაფიკული ხელსაწყოს - Expression Blend-ის გამოყენება.

Silverlight ტექნოლოგიაზე პროგრამული უზრუნველყოფის შემუშავებისას რეკომენდირებულია MVVM არქიტექტურული სტანდარტის გამოყენება, რომელიც Silverlight ტექნოლოგიით შემოთავაზებული შესაძლებლობების სრულად რეალიზაციის საშუალებას იძლევა. MVVM მოდელი შედგება სამი ძირითადი ლოგიკური კომპონენტისგან:

- Model: ბიზნეს დომენი (ბიზნეს მოთხოვნები, მონაცემთა წვდომის ლოგიკა, model-კლასები).
- View: სამომხმარებლო ინტერფეისი (Silverlight screens).
- ViewModel: შუალედური დონე View-სა და Model- დონეებს შორის.



ნახ. 6 MVVM არქიტექტურული სტანდარტი

მეორე თავში გადმოცემულია სერვისზე ორიენტირებული პროგრამული უზრუნველყოფის არქიტექტურა და მისი დაპროექტების რეკომენდაციები.

პროგრამული დანართის არქიტექტურა არის სტრუქტურიზებული გადაწყვეტილების მიღების პროცესი, რომელიც აკმაყოფილებს აპლიკაციის მიმართ წამოყენებულ ყველა ტექნიკურ და ფუნქციონალურ მოთხოვნილებას, ამასთან ერთად მიღწეულია მაღალი ხარისხობრივი მახასიათებლები, როგორიცაა უსაფრთხოება, მართვადობა და სისწრაფე.

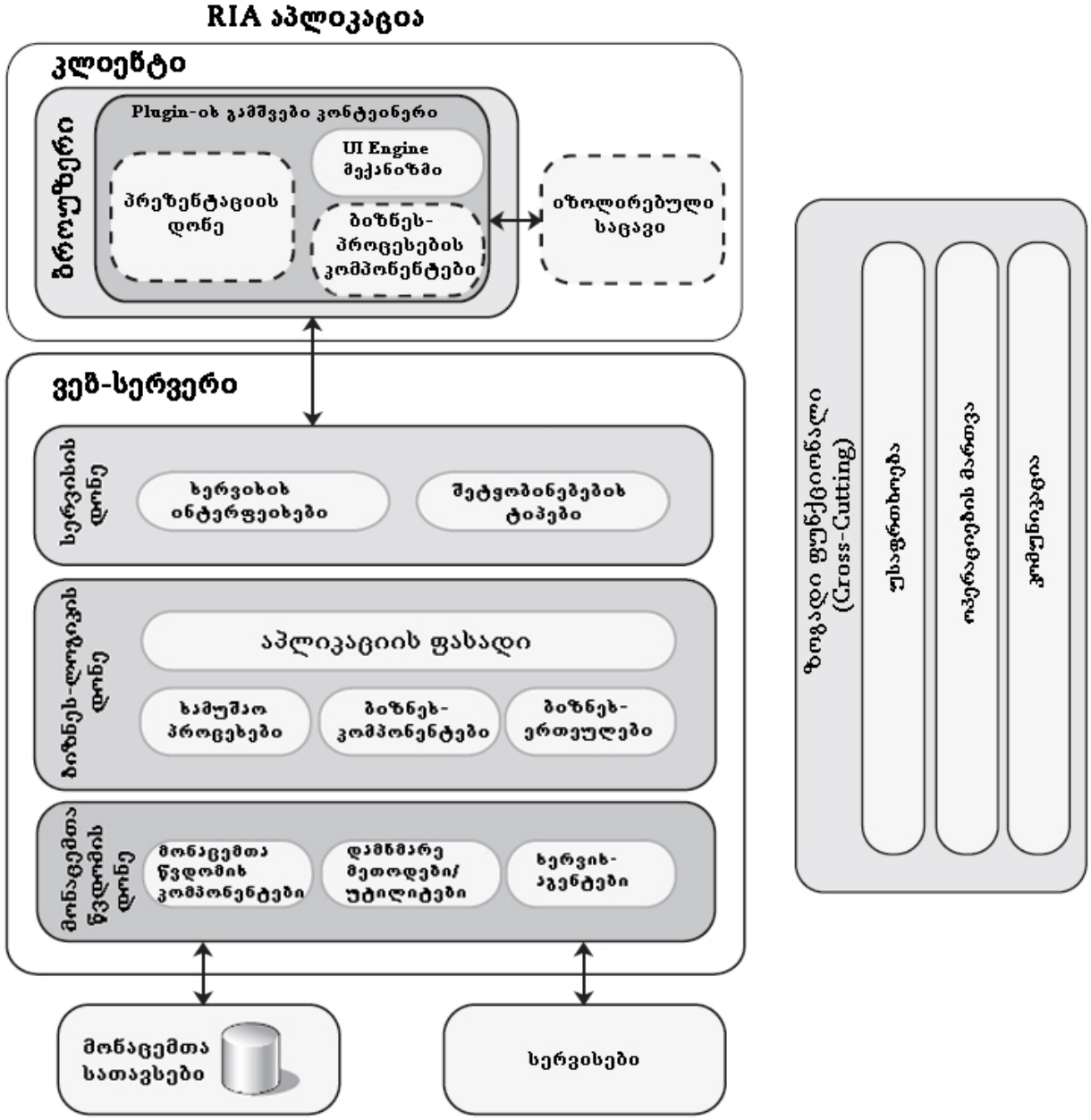
სხვა ნებისმიერი სტრუქტურის მსგასად, პროგრამული დანართის წარმატებაში მნიშვნელოვან როლს თამაშობს მყარი საფუძველი. პროგრამული დანართი უნდა დაიგეგმოს მომხმარებელის, სისტემისა (IT ინფრასტრუქტურა) და ბიზნეს-მოთხოვნილებებთან თანხვედრაში. პროგრამული დანართის დაგეგმვისას მხედველობაში უნდა ვიქონიოთ სამომხმარებლო არეალი, პოტენციური მომხმარებლის მოთხოვნები, რეალურ სერვერზე დანერგვისა და შემდგომი მხარდაჭერის საკითხები.

დისერტაციაში წარმოდგენილია პროგრამული დანართის ლოგიკურ დონეებზე გადანაწილებული არქიტექტურა, რაც გულისხმობს ერთმანეთთან დაკავშირებული ფუნქციონალის ცალკეულ ლოგიკურ შრეებში გაერთიანებას. ეს შრეები ერთი-მეორეზე ვერტიკალურად არის დალაგებული. თითოეული შრის ფუნქციონალი ატარებს რაიმე ერთ კონკრეტულ დანიშნულებას და შრეებს შორის მიმდინარეობს ინფორმაციის მიმოცვლა, კომუნიკაცია. პროგრამული დანართის გადანაწილება ლოგიკურ შრეებში თავიდან გვარიდებს კონცეფციების მჭიდრო ურთიერთ-გადაჯაჭვულობას, უზრუნველყოფს მოქნილობას და იოლად მართვას.

ლოგიკური შრეების არქიტექტურა მუშაობის პრინციპით წააგავს ამობრუნებულ პირამიდას: ყოველი შრე უკავშირდება და იყენებს მის უშუალოდ ერთი დონით ქვემოთ მდგომი შრის ფუნქციონალს. შრეების უფრო თავისუფალ არქიტექტურაში დაშვებულია ნებისმიერ ქვედა დონის შრესთან ურთიერთქმედება.

დისერტაციაში წარმოდგენილია Silverlight ტქექნოლოგიით დაწერილი RIA აპლიკაციის არქიტექტურა და კომპონენტების გადანაწილება ლოგიკურ შრეებში. RIA აპლიკაცია უზრუნველყოფს მდიდარ გრაფიკულ სამომხმარებლო ინტერფეისს და Streaming Media სერვისებს, ამავდროულად, გააჩნია ვებ-აპლიკააცისთვის დამახასიათებელი თვისებები: იოლად განთავსებადია და მართვადი (Deployment პროცესი, პროგრამული უზრუნველყოფის შემდგომი მხარდაჭერა). RIA-აპლიკაციები ეშვება ბრაუზერის Plug-In გარემოში, რითაც განსხვავდება სხვა ვებ-ტექნოლოგიებისგან, რომლებიც იყენებენ ბრაუზერის კოდს (მაგ. AJAX ტექნოლოგია). ტერმინი Plug-In აღწერს კომპიუტერულ პროგრამას, რომელიც ხელს უწყობს სხვა პროგრამული დანართების მუშაობის გაუმჯობესებას ან ფუნქციონალის გამდიდრებას.

ტიპიური RIA-იმპლემენტაცია ვებ-ინფრასტრუქტურას იყენებს კლიენტის-მხარის აპლიკაციასთან ერთობლიობაში, რომელიც პრეზენტაციის დონეს უზრუნველყოფს. ბრაუზერის Plug-In დანამატი იძლევა მდიდარ გრაფიკულ ინტერფეისთან სამუშაო ბიბლიოთეკებს და კონტეინერის როლს თამაშობს, რათა უსაფრთხოების დაცვის მიზნით, ლოკალურ რესურსებზე შეზღუდოს წვდომა. RIA-აპლიკაციებს ახასიათებს უფრო კომპლექსური და მდიდარი სამომხმარებლო გარემო (კლიენტის მხარე, Client-Side), ვიდრე შესაძლებელი იყო ტრადიციული ვებ-აპლიკაციებისთვის, რის შედეგადაც ვებ-სერვერზე დატვირთვა მსუბუქდება და ნაწილდება კლიენტის მანქანაზე. ნახ.5-ზე ნაჩვენებია RIA-აპლიკაციის სტრუქტურა:



ნახ. 7 RIA აპლიკაციის არქიტექტურა.
წყვეტილი ხაზები აღნიშნავს არასავალდებულო კომპონენტებს.

ძირითადი სტანდარტები (Design Pattern) ორგანიზებულია კატეგორიებად, სექციებში: შრეები (Layer), კომუნიკაცია (Communication), კომპოზიცია (Composition), პრეზენტაცია (Prezentation). ამ სტანდარტების გათვალისწინებით ვხელმძღვანელობთ არქიტექტურული გადაწყვეტილებების არჩევისას:

კატეგორია	შესაბამისი სტანდარტი (Design Pattern)
შრეები (Layers)	**Service Layer.** არქიტექტურული სტანდარტი, რომელშიც სერვისის ინტერფეისი და იმპლემენტაცია გაერთიანებულია ერთ ლოგიკურ შრეზე. სერვისი იმპლემენტირებულია WCF ტექნოლოგიით.
კომუნიკაცია Communication)	**Asynchronous Callback.** გრძელვადიანი სამუშაოების განცალკევებულ Background Thread ფონზე გაშვება და Call Back ფუნქციით შესრულებული სამუშაოს შედეგების ამოკითხვა. **Command.** მოთხოვნის დამუშავების ლოგიკის ინკაფსულაცია command ობიექტში, რომელიც იძლევა Common Execution Interface გარემოს.
კომპოზიცია (Composition)	**Compozite View.** ინდივიდუალური View გვერდების გაეთიანება კომპოზიტურ View-ში. **Inversion of Control.** ობიექტების სხვა ობიექტებზე/კომპონენტებზე დამოკიდებულებების აღწერა, რომელთა იმპლემენტაცია სავალდებულოა, რათა მოცემული ობიექტის გამოყენება შესაძლებელი გახდეს პროგრამულ დანართში.
პრეზენტაცია (Prezentation)	**Application Controller.** ობიექტი, რომელიც შეიცავს სრული სამუშაო პროცესის ლოგიკას და გამოიყენება სხვა კომპონენტების მიერ, რომლებიც მუშაობენ Model ობიექტთან და ასახავენ შესაბამის View პრეზენტაციას. **Supervising Presenter.** პრეზენტაციის დონის სტრუქტურის გადანაწილება სამ დამოუკიდებელ როლში: View პასუხისმგებელია მომხმარებლთან ურთიერთ-კავშირზე (User Input); მონაცემთა ბმის მექანიზმით გამოაქვს Model კომპონენტის ინფორმაცია, ხოლო Model კომპონენტში ინკაფსულირებულია ბიზნეს-ლოგიკა; Presenter ობიექტში იმპლემენტირებულია პრეზენტაციის ლოგიკა და კოორდინირებას უკეთებს View-სა და Model-ობიექტებს შორის ურთიერთქმედებებს. **Presentation Model.** წარმოადგენს Model-View-Presenter (MVP) არქიტექტურული სტანდარტის ვარიაციას და შექმნილია თანამედროვე სამომხმარებლო ინტერფეისების დეველოპმენტ- პლატფორმებისთვის, სადაც View გრაფიკული დიზაინერის დანიშნულებას უფრო ატარებს, ვიდრე პროგრამული ლოგიკის.

ცხრილი 1. RIA-აპლიკაციებში გამოყენებული არქიტექტურული სტანდარტები

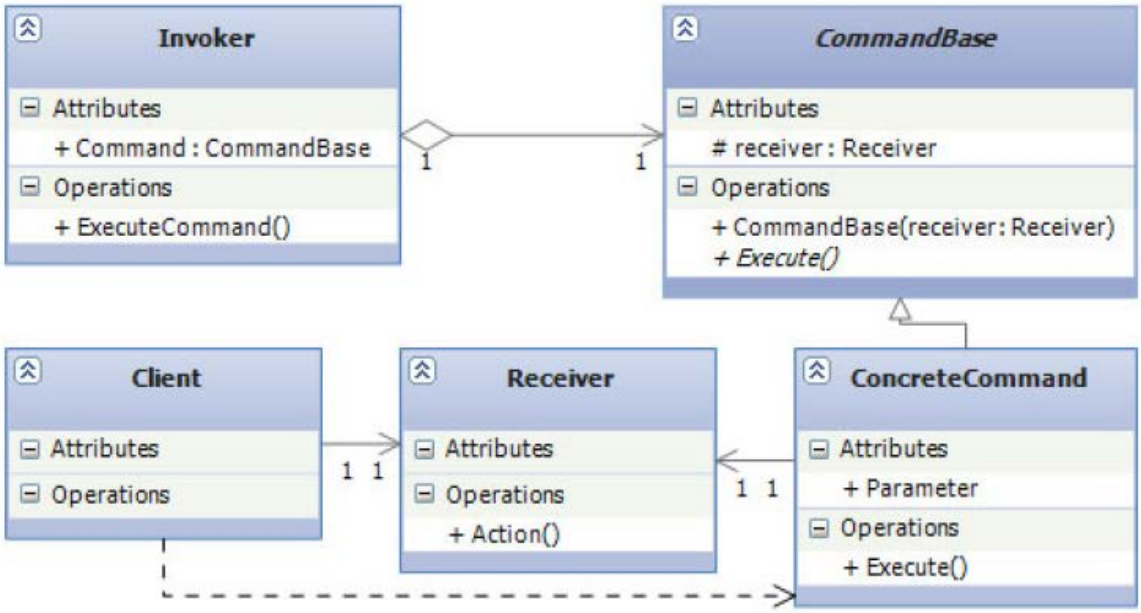
Silvelight აპლიკაციებში მნიშვნელოვან როლს თამაშობს Command არქიტექტურა. ეს სტანდარტი შემომავალ მოთხოვნებზე საპასუხო ლოგიკას აერთიანებს ერთ ობიექტში. Command ობიექტის გამოძახება მოთხოვნის შემოსვლისთანავე ხდება. ამ არქიტექტურული მიდგომის მთავარი დამახასიათებელია ის, რომ ამა თუ იმ პროგრამული ლოგიკის შესასრულებლად საჭირო ინფორმაცია თავმოყრილია ერთ ობიექტში. თავად ობიექტი არ ასრულებს რაიმე ოპერაციას, იგი მხოლოდ ინფორმაციას შეიცავს.

განვმარტოთ სამი მნიშვნელოვანი ცნება: კლიენტი (Client), გამომძახებელი (Invoker) და მიმღები (Receiver). კლიენტი ქმნის Command ობიექტს. გამომძახებელი წყვეტს თუ როდის უნდა მოხდეს Command ობიექტში აღწერილი მეთოდის გამოძახება. მიმღები კი არის კლასის ეგზემპლიარი, რომელიც შეიცავს ამ მეთოდის პროგრამულ კოდს.

ნახ.6-ზე მოყვანილია UML დიაგრამა, რომელიც აღწერს Command არქიტექტურული სტანდარტის იმპლემენტაციას. ეს დიაგრამა შედგება 5 ნაწილისგან:

- კლიენტი (Client): კლასი, რომელიც წარმოადგენს Command არქიტექტურის მომხმარებელს. იგი ქმნის Command ობიექტს და აკავშირებს მიმღებთან (Receiver).
- მიმღები (Receiver): კლასი, რომელმაც იცის თუ როგორ უნდა განახორციელოს შემომავალ მოთხოვნასთან დაკავშირებული ოპერაციები.
- CommandBase: აბსტრაქტული კლასი (ან ინტერფეისი) ყველა Command ობიექტისთვის. იგი შეიცავს ინფორმაციას თუ რომელი მიმღები (Receiver) არის პასუხისმგებელი Command ობიექტში ინკაპსულირებული ოპერაციების შესრულებაზე.
- კონკრეტული იმპლემენტაცია (ConcreteCommand): CommandBase აბსტრაქტული კლასის (ან ინტერფეისის) კონკრეტული იმპლემენტაცია.
- გამომძახებელი (Invoker): ობიექტი, რომელიც წყვეტს თუ როდის უნდა გეშვას Command ობიექტი.

Command არქიტექტურული სტანდარტი Silverlight ვებ-აპლიკაციებში გვეხმარება ხდომილებების შეზღუდვების დაძლევაში, რადგან გრაფიკულ სამომხმარებლო გარემოში (View) აღიძვრება Command ობიექტი და ViewModel შრეზე ხდება მისი მიღება, რომელმაც იცის რა სამუშაოები უნდა ჩაატაროს საპასუხოდ.



ნახ. 8. Command არიტექტურული სტანდარტის იმპლემენტაცია

Command სტანდარტის იმპლემენტაცია MVVM არქიტექტურაში

I. ICommandSource ინტერფეისი

ბრძანების გამომძახებელი ობიექტი (Invoker) წარმოადგენს ICommandSource ინტერფეისის წევრს. ქვემოთ მოცემულია IcommandSource ინტერფეისის პროგრამული კოდი:

```
public interface ICommandSource
{
    ICommand Command { get; }
    object CommandParameter { get; }
    IInputElement CommandTarget { get; }
}
```

ამ არქიტექტურით ნათლად ჩანს, რომ ყოველი კლასი, რომელიც იმპლემენტაციას უკეთებს ICommandSource ინტერფეისს, უნდა შეიცავდეს მხოლოდ ერთ Command ობიექტს. თავად Command ობიექტი წარმოადგენს ICommand ინტერფეისის იმპლემენტაციას. Silverlight პროგრამულ უზრუნველყოფაში კონტროლები Command ობიექტს ამჟღავნებენ DependencyProperty სახით, რათა მონაცემთა ბმით შევძლოთ მისი მართვა. CommandParameter წამოადგენს ობიექტს, რომელიც ინდივიდუალური Command იმპლემენტაციისთვის სახასიათო ინფორმაციას შეიცავს და ნებისმიერი ტიპის მნიშვნელობის მიღება შეუძლია.

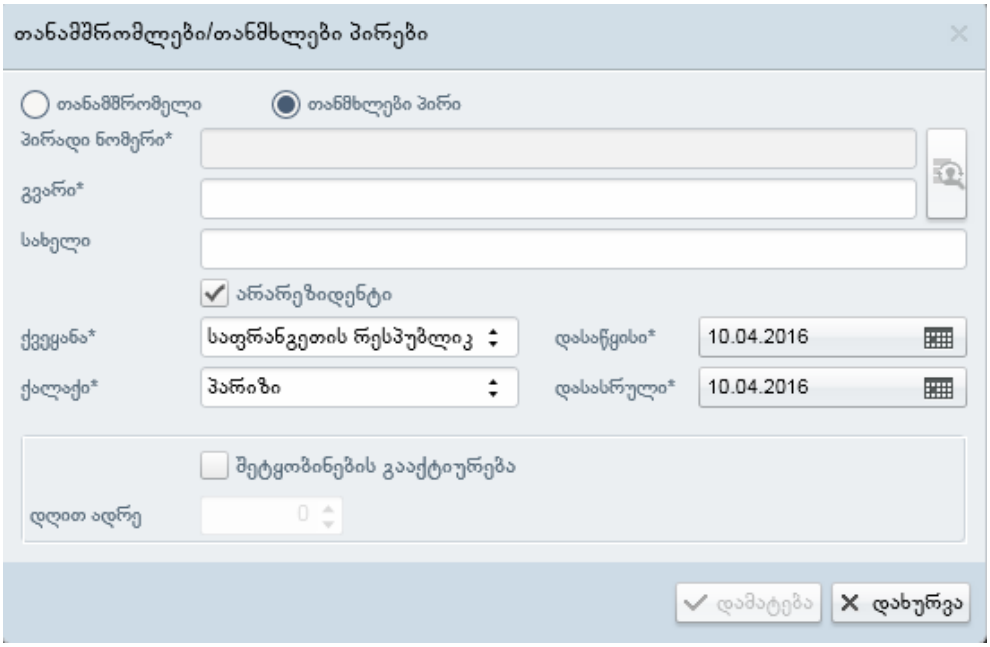
II. ICommand ინტერფეისი

MVVM არქიტექტურაში ნებისმიერი ბრძანება წარმოადგენს ICommand ინტერფეისის იმპლემენტაციას. იგი აღწერს კონტრაქტს, რომელიც Command ობიექტის თითოეულმა იმპლემენტაციამ უნდა დააკმაყოფილოს. ICommand ინტერფეისის აღწერა მოყვანილია ქვემოთ:

```
public interface ICommand
{
    event EventHandler CanExecuteChanged;
    bool CanExecute(object parameter);
    void Execute(object parameter);
}
```

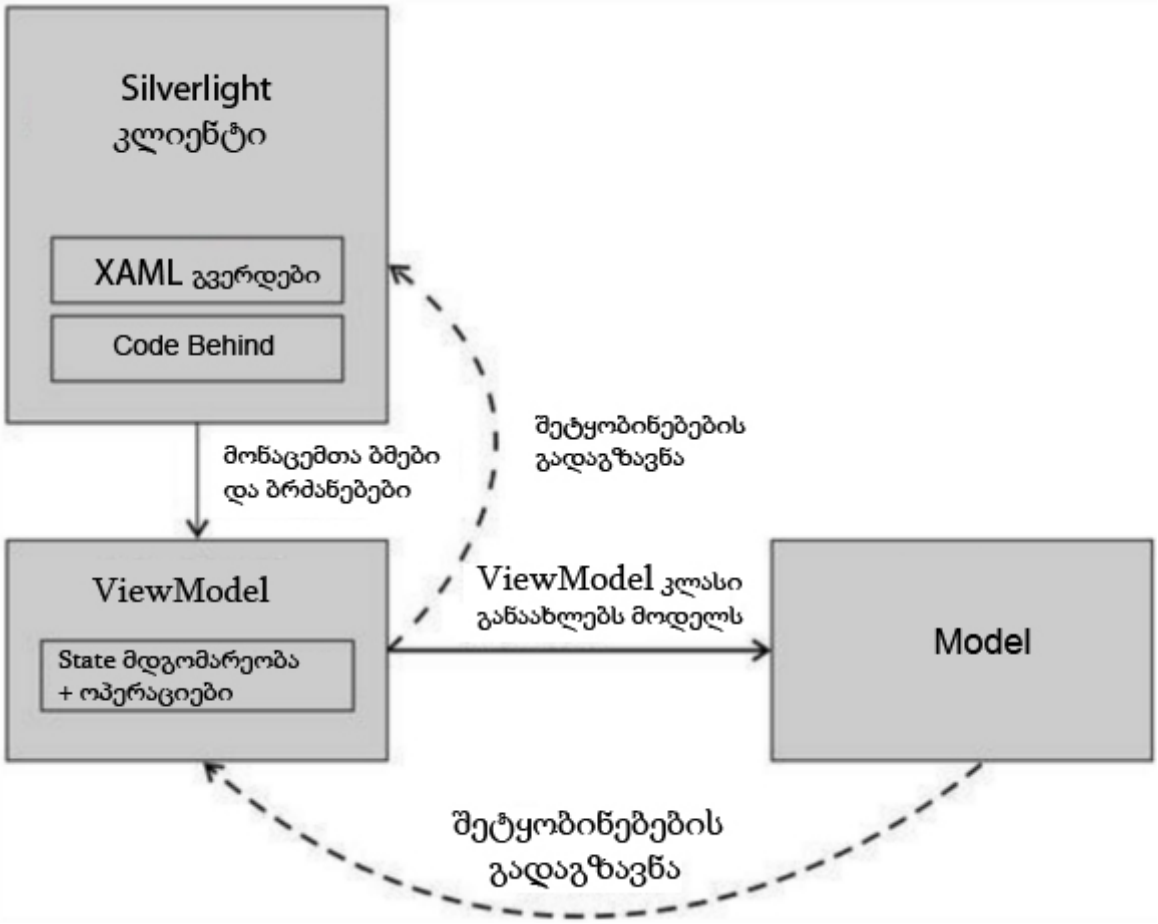
მეთოდი Execute წარმოადგენს Command აქრიტექტურის მთავარ ნაწილს. იგი ინკაპსულირებას უკეთებს Command ობიექტის მიერ შესასრულებელ სამუშაოს. Execute მეთოდს ინდივიდუალური კონტექსტიდან ნებისმიერი ტიპის პარამეტრი შეგვიძლია გადავცეთ (რადგან C#-ში ყველა ობიექტი მემკვიდრეობით მოდის object ტიპიდან).

CanExecute მეთოდი გამომძახებელს ატყობინებს დაშვებულია თუ არა მოცემულ მომენტში მითითებული ბრძანების შესრულება. პრაქტიკულად იგი გამოიყენება სამომხმარებლო ინტერფეისზე გამომძახებელი ობიექტის (მაგ. ღილაკი) ჩართვისა და ამორთვის მიზნით (Enable, Disable).



ნახ. 9 ღილაკი "დამატება" ჩამქრალია, რადგან ფორმაზე არ არის შეტანილი სავალდებულო ინფორმაცია

სადისერტაციო ნაშრომში ასევე განხილულია პრეზენტაციის დონეზე მუშაობის სქემა, კერძოდ, View და ViewModel დონეებს შორის კომუნიკაცია. პრეზენტაციის დონე აღწერს სამომხმარებლო ინტერფეისს. ყოველი გვერდი code-behind კლასი შეიცავს პროგრამული კოდის მინიმალურ რაოდენობას, და ისიც, მხოლოდ იმ შემთხვევაში, თუ გრაფიკული ინტერფეისის მართვის გარკვეული როლი აქვს დაკისრებული. ძირითადი პროგრამული ლოგიკა იმპლემენტირებულია ViewModel კლასში, რომელიც ინახავს აპლიკაციის მდგომარეობას (State) და პრეზენტაციის დონეზე აგზავნის შეტყობინებებს. ეს დონეები ერთმანეთთან შემდეგი სქემით უქრთიერთქმედებენ:



ნახ. 10. Silverlight კლიენტის, ViewModel და Model დონეებს შორის ურთიერთქმედება

View გვერდის კონტროლები მონაცემთა ბმით დაკავშირებულია ViewModel კლასის წევრებთან (Properties), როგერც წესი - დეკლარატიული გზით. ViewModel კლასში აღწერილია ბრძანებების სიმრავლე, როგორიცაა ცვლილებების შენახვა მონაცემთა სათავსოში. პრეზენტაციის დონე მათ კონკრეტულ კონტროლებთან აკავშირებს, რათა მომხმარებელს ქმედების

განხორციელების საშუალება მიეცეს. მაგალითად, მომხმარებელს შეუძლია Save ღილაკზე დაჭერა, რომელიც ViewModel კლასის SaveDataCommand ბრძანებაზე არის მიბმული.

ამას გარდა, ViewModel დონე ინფორმაციის მისაღებად მოთხოვნებს უგზავნის Model დონეს. მოდელის დონე იწყებს გამოთვლით პროცესებს და როდესაც დაასრულებს სამუშაოებს, ViewModel ობიექტს გადაუგზავნის შეტყობინებებს (ამ შემთხვევაში იგულისხმება, რომ მოდელი კლიენტის მხარეს იმყოფება და სერვერულ მხარეს მიმართავს ოპერაციების შესასრულებლად - ინფორმაციის წამოღება ან ცვლილება). ViewModel ობიექტი შედეგების მიღების შემდეგ გადააგზავნის შეტყობინებებს View გვერდის მიმართულებით და მონაცემთა ბმის მექანიზმით განაახლებს სამომხმარებლო ინტერფეისს.

სადისერტაციო ნაშრომში შემოთავაზებულია ანგარიშგებების მოდულის ინტეგრაციის მეთოდები პროგრამულ უზრუნველყოფაში. მონაცემთა ანალიზისა და ანგარიშგებების გენერაციის მიზნით გამოყენებულია კომპანია Microsoft-ის ხელსაწყო SQL Server Reporting Services (SSRS).

SQL Server Reporting Services წარმოადგენს სერვერზე დაფუძნებულ, განვრცობად პლატფორმას, რომელშიც ხდება ინფორმაციის დამუშავება, ფორმატირება და მომხმარებლისთვის ტრადიციული თუ ინტერაქტიული ფორმატით მიწოდება. SSRS კონფიგურირებადია და ხასიათდება მრავალი ფუნქციით, როგორიცაა მაგალითად, მონაცემთა ვიზუალიზაცია დიაგრამებისა და გრაფიკების სახით, ანგარიშგებათა სხვადასხვა ფორმატით ექსპორტირება (HTML, Excel, PDF, მომხმარებლის ელ-ფოსტაზე გადაგზავნა, დაკონფიგურირება და SharePoint კორპორატიულ პორტალებში ჩაშენება.

ნაშრომში წარმოდგენილია პროგრამულ უზრუნველყოფაში ანგარიშგებების ჩაშენების სამი შესაძლო ვარიანტი:

- Reporting Server Web Service (ასევე ცნობილი როგორც Reporting Services SOAP API)
- ReportViewer კონტროლი
- წვდომა URL გზავნილის საშუალებით

მესამე თავში გადმოცემულია დისერტაციაში განხილული თეორიული საკითხების პრაქტიკული რეალიზება. განიხილება ადამიანური რესურსების მართვის ელექტრონული სისტემა (eHRMS). იგი აღწერს HRM დისციპლინას, ორგანიზაცულ სტრუქტურას, თანამდებობების იერარქიულ ხეს და ძირითად ოპერაციებს ადამიანურ რესურსებზე. მთლიანობაში, HRMS სისტემის ზოგადი დანიშნულებაა სხვა და სხვა პროგრამული უზრუნველყოფიდან თავი მოუყაროს ინფორმაციას და ერიან, უნივერსალურ მონაცემთა სათავსოში გააერთიანოს. შექმნილი პროგრამული უზრუნველყოფა წარმატებით გამოიყენება ორმოცზე მეტ სახელმწიფო ორგანიზაციაში.

სისტემა აკმაყოფილებს მის მიმართ წაყენებულ ზოგად მოთხოვნებს და შეიცავს მომხმარებლისთვის საჭირო ფუნქციონალს. იგი წარმოადგენს მოქნილ და დინამიურ სისტემას, დანერგვის შემდეგაც, განუწყვეტლივ მიმდინარეობს მისი განვითარება, ცვლილებები და განახლება.

HRMS პროგრამული უზრუნველყოფა დაწერილია .NET Framework 5.0 პროგრამულ გარემოში, C# ენაზე, SQL Server, WCF, Silverlight, SQL Server Reporting Services პროგრამული პაკეტების გამოყენებით.

ელექტრონულ მართვის სისტემაში ძირითად რგოლს წარმოადგენს თანამშრომელი, რომლის შესახებ რეგისტრირდება ინფორმაცია, როგორც მანუალურად, ასევე ავტომატურად, მოდულებს შორის ინფორმაციის გაცვლის საშუალებით (სერვისების გამოძახება).

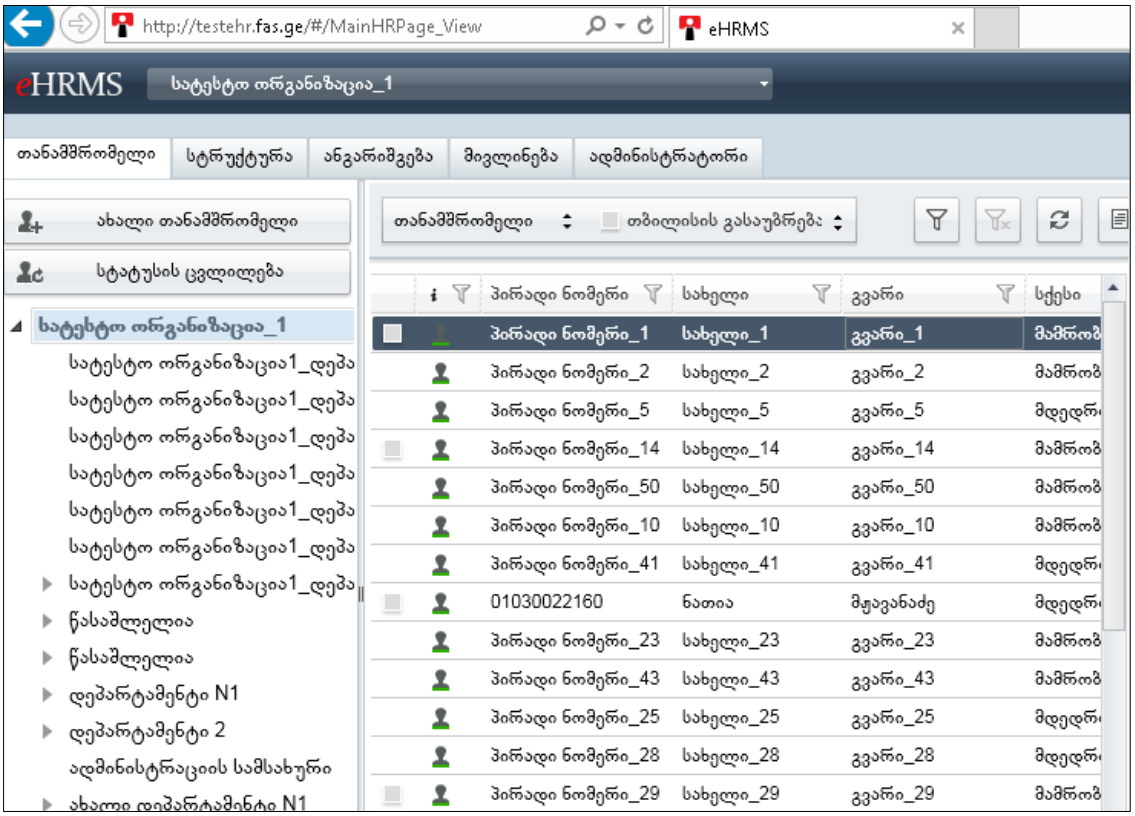
ყველა თანამშრომელი სისტემაში რეგისტრირდება ინდივიდუალურად და ხედავს თავის სამუშაო გარემოს, სისტემაზე დაშვებები რეგულირდება შიდა მოხმარების ნორმატიული აქტებით.

ელექტრონული მართვის სისტემა მოიცავს ცხრილ 2-ში აღწერილ მოდულებს. მოდულის საშუალებით თითოეული მომხმარებელი, თანამდებობრივი ჯგუფის (დონის) შესაბამისად იღებს ინფორმაციას და ასევე ასრულებს მოქმედებებს.

N	მოდული	აღწერა
1	ა.) ორგანიზაციის სტრუქტურა. ბ.) თანამდებობების სტრუქტურა (დაქვემდებარების იერარქიული ხე)	ორგანიზაციაზე, სტრუქტურულ ერთეულზე ან თანამდებობაზე განხორციელებული მოქმედებების აღრიცხვა. ორგანიზაციის სტრუქტურის აგება, ცვლილება.
2	თანამშომელთა მართვა (პიროვნების ბარათი).	თანამშრომლის ბარათზე განხორციელებული მოქმედებების აღრიცხვა. თანამშრომლის დანიშვნა, გადანიშვნა, გათავისუფლება, თანამდებობების შეთავსება. თანამშრომლებზე ინფორმაციის მუდმივი განახლება.
3	ახალი თანამშრომელის დაქირავება.	ვაკანსიის გამოცხადება, გადაწყვეტილების ასახვა. სისტემაში ახალი თანამშრომლის დარეგისტრირება და თანამდებობაზე დანიშვნა.
4	თანამშრომლების დასწრების და საშვების აღრიცხვა.	არსებული დაშვების სისტემასთან ინტეგრაცია და საშვების აღრიცხვა.
5	სასწავლო ცენტრის მართვის, ატესტაციებისა და ტრეინინგების მოდული.	შეხვედრების რეზერვირება და დასწრების და შედეგების ასახვა.
6	თანამშრომელთა მიერ დახარჯული სამუშაო დროის აღრიცხვის მოდული.	მხოლოდ აუდიტების მიერ შესავსები ფორმების შევსება.
7	სახელფასო პროგრამა, სტანდარტული მინიმალური ფუნქციონალით.	მხოლოდ სახელფასო სიის მომზადება (სტანდარტული ფუნქციონალი).
8	პერიოდული და მიმდინარე ანგარიშების (რეპორტების) მომზადებისა და ბეჭდვის მოდული.	წინასწარ მომზადებული შაბლონების მიხედვით ანგარიშების მომზადება.
9	სისტემის ადმინისტრირების მოდული.	სისტემის მართვა, მომხმარებლების წვდომის ადმინისტრირების პანელი.
10	შემოსული, გასული და შიდა წერილების აღრიცხვის (კანცელარიის) მოდული	შემოსული გასული წერილების რეესტრი და მარშუტის და რეზოლუციების ისტორია

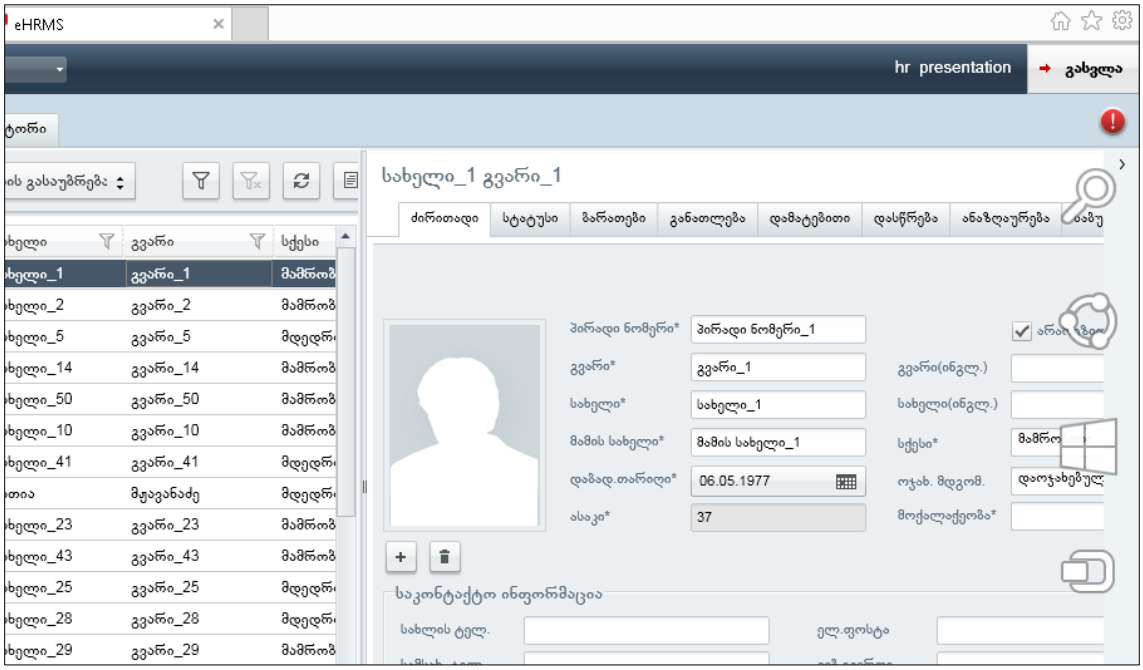
ცხრილი 2. ელექტრონული კადრების მართვის სისტემის მოდულები

თანამშრომლის პირადი ბარათი. ძირითადი ინფორმაცია თანამშრომლის შესახებ განთავსებულია პირად ბარათზე და კლასიფიცირებულია ჩანართებში. საწყის გვერდზე გამოტანილია სისტემაში დარეგისტრირებული თანამშრომლების ნუსხა. თანამშრომლის ჩანაწერის მონიშვნის შემდეგ (ლურჯი ფერით გამოკვეთილი სტრიქონი) გააქტიურდება თანამშრომლის პირადი ბარათის ჩანართი. პირად ბარათზე დატანილია თანამშრომლის შესახებ ძირითადი ინფორმაცია (ფოტოსურათი, პირადი ნომერი, საკონტაქტო ინფორმაცია, ფაქტიური და იურიდიული მისამართი, მოქალაქეობა, ინფორმაცია ნასამართლეობისა და ჯანმრთელობის მდგომარეობის შესახებ და ა.შ.), გარდა ძირითადი ინფორმაციის ტაბელისა, თანამშრომლის პირად ბარათზე დატანილია დამატებითი ინფორმაცია, რომელიც ლოგიკურად გადანაწილებულია შესაბამის ტაბებში: „სტატუსი“, „ბარათები“, „განათლება“,



ნახ. 11 თანამშრომლების სია.

„დასწრება“, „ანაზღაურება“ და სხვა. ქვემოთ მოყვანილ სურათზე ილუსტრირებულია აპლიკაციის საწყისი გვერდი.



ნახ. 12 თანამშრომლის პირადი ბარათი. ძირითადი ინფორმაციის ჩანართი.

დასკვნა

სადისერტაციო თემის ფარგლებში ჩატარებული საპროექტო–კვლევითი სამუშაოების შედეგების საფუძველზე შესაძლებელია შემდეგი დასკვნების გაკეთება:

1. სადისერტაციო ნაშრომში „კორპორაციული Web-აპლიკაციების აგების ტექნოლოგიები სერვის-ორიენტირებული არქიტექტურით“ განხილულია პროგრამული უზრუნველყოფის შექმნის მეთოდოლოგია, კომპანია Microsoft-ის პროგრამული პლატფორმების გამოყენებით (ASP.NET Web Forms, ASP.NET MVC და Silverlight). ყურადღება გამახვილებულია დაპროგრამების ტექნოლოგიების მახასიათებლებზე, შესაბამისი პრაქტიკული ექსპერიმენტების საფუძველზე გამოვლენილია მათი დადებითი და უარყოფითი მხარეები. შემოთავაზებულია რეკომენდაციები, რეალურ ამოცანებზე მუშაობისას, რა შემთხვევებში აჯობებს დაპროგრამების ამა თუ იმ ტექნოლოგიის არჩევა.
2. განხილულ იქნა პროგრამული უზრუნველყოფის ზოგადი არქიტექტურა, დაპროექტების დროს გასათვალიწინებელი რეკომენდაციები და რჩევები. ზოგადი სტანდარტების გათვალიწინებით შემუშავდა ახალი ტექნოლოგიის (Silverlight) გამოყენების დროს RIA აპლიკაციების არქიტექტურა, რომელიც გულისხმობს MVVM სტანდარტზე აგებულ პროგრამულ ლოგიკას. გამოვლენილია პროგრამული დანართის ლოგიკურ კომპონენტებს შორის კომუნიკაციის მეთოდოლოგია.
3. Silverlight ვებ-აპლიკაციების შემუშავებისას კარგ შედეგს იძლევა MVVM არქიტექტურის გამოყენება. პროგრამული ლოგიკის პრეზენტაციის დონეზე (.XAML ფაილის უკანა მხარეს, code-behind) წერის ნაცვლად ლოგიკამ გადანაცვლა ViewModel კლასებში. ეს მიდგომა უფრო სრულყოფილია და Silverlight ტექნოლოგიით შემოთავაზებული შესაძლებლობების სრულად რეალიზაციის საშუალებას იძლევა.
4. არქიტექტურული სტანდარტების გამოყენება მაღალი ხარისხის შედეგს იძლევა Silverlight პროექტებში. სერვისის დონეზე რეალიზებულია WCF ტექნოლოგიისა და პროქსი კლასებით. ასინქრონული მეთოდების გამოძახებები და Command არქიტექტურა წარმატებით მუშაობს პრეზენტაციის დონეზე. ასევე, დადებით შედეგს იძლევა კომპოზიტური გვერდების გამოყენება (ინდივიდუალური View გვერდების გაეთიანება კომპოზიტურ View-ში).
5. ანგარიშგებების მოდულის ჩაშენება პროგრამულ უზრუნველყოფაში ინფორმაციის გაანალიზებისა და სტატისტიკური მაჩვენებლების დათვლის საშუალებას იძლევა. უსაფრთხოების საკითხს მნიშვნელოვანი როლი აკისრია - მხოლოდ ავტორიზებულ მომხმარებლებს უნდა ჰქონდეთ დაშვება ინფორმაციაზე. გარდა ამისა, მომხმარებლებში აქტუალურია ანგარიშგებების ექსპორტირების შესაძლებლობა სხვა და სხვა ფაილურ ფორმატში, ასევე, დოკუმენტების გენერაცია შაბლონის საფუძველზე.

6. Silverlight ტექნოლოგიის გამოყენებით მიიღწევა მდიდარი სამომხმარებლო გარმოს შემუშავება. RIA აპლიაციები უფრო მაღალი დონის გარემოს სთავაზობს მომხმარებელს, ვიდრე ტრადიციული ვებ-აპლიკაციები. RIA აპლიკაცია ეშვება ვებ-ბრაუზერში და თავსებადია სხვადასხვა პლატფორმასა და ვებ-ბრაუზერთან.

7. სამომხმარებლო გარემოს გამდიდრება Silverlight პლატფორმის კონტროლებით. Silverlight ტექნოლოგიის მნიშვნელოვანი უპირატესობა არის მძლავრი მონაცემთა ბმის მექანიზმი და კონვერტაციის საშუალება. Value Converter ობიექტი სამომხმარებლო ინტერფეისზე გამოტანილი ინფორმაციის ფორმატირებისა და ტრანსფორმირების საშუალებას იძლევა.

8. Silverlight ტექნოლოგიის გამოყენება კიდევ ერთ უპირატესობას იძლევა: სამომხმარებლო ინტერფეისი მნიშვნელოვნად შეგვიძია გავაუმჯობესოთ ვებ აპლიკაციებში ანიმაცების დამატებით. პროგრამისტს აღარ ჭირდება დამატებით სხვა ტექნოლოგიებთან მუშაობა (Adobe Flash), Expression Blend ხელსაწყოში გათვალისწინებულია ანიმირებისა და ტრანსფორმაციის ეფექტების შექმნა. Silverlight ტექნოლოგია შეიცავს შესაბამის ბიბლიოთეკებს. ანიმაციის ყველა ტიპი Silverlight ტექნოლოგიაში მემკვიდრეობით მოდის Timeline კლასიდან, რომელიც განთავსებულია System.Windows.Media.Animation ბიბლიოთეკაში.

გამოქვეყნებული ლიტერატურა

1. **კივილაძე ნ.**, სურგულაძე გ., გულიტაშვილი მ. (2015). Microsoft-ის Web-ტექნოლოგიების ანალიზი და განვითარების ტენდენციები საინფორმაციო სისტემებისთვის. სტუ-ს შრ.კრ. „მას“. No 1(17), თბილისი, გვ.84-87
2. სურგულაძე გ., **კივილაძე ნ.** (2015). Web-აპლიკაციაში რეპორტების ინტეგრაციის მეთოდები. თეზისი, VII საერთაშორისო სამეცნიერო და პრაქტიკული კონფერენცია „ინტერნეტი და საზოგადოება” (INSO2015). ქუთაისი. 2015. გვ.92-96
3. **კივილაძე ნ**. (2015). რევოლუცია ASP.NET პლატფორმაზე: Web API და AngularJS ტექნოლოგიების გამოყენებით ერთგვერდიანი აპლიკაციების შემუშავება. აკად. ი. ფრანგიშვილის დაბ. 85 წ. მიძღვ. საერთაშ.სამეცნ. კონფ.: „საინფორმაციო და კომპიუტერული ტექნოლოგიები, მოდელირება, მართვა“, შრომები. თბილისი. გვ.229-233
4. სურგულაძე გ., გულიტაშვილი მ., **კივილაძე ნ.** (2015). Web-აპლიკაციების ტესტირება, ვალიდაცია და ვერიფიკაცია. მონოგრაფია. ISBN 978-9941-0-7682-4. სტუ-ს „IT-კონსალტინგის ცენტრი”. თბილისი. 205 გვ.
5. სურგულაძე გ., თოფურია ნ., ბასილაძე გ., კივილაძე ნ., ნეფარიძე მ. (2014). ელექტრონული საარჩევნო სისტემა მულტიმედიური მონაცემთა ბაზებით და კლიენტ-სერვერ არქიტექტურით. GESJ: Computer Science and Telecommunications, 2014, N2(42). გვ.39-86. /იმპაქტ ფატორი N0.8125/
6. **კივილაძე ნ.,** გულიტაშვილი მ. (2014). Web-აპლიკაციების დეველოპმენტის და ტესტირების ტექნოლოგიები საინფორმაციო სისტემებისთვი. III საერთაშ. სამეცნ. კონფ.: „კომპიუტინგი / ინფორმატიკა, განათლების მეცნიერებები“. ბათუმი, გვ.42
7. სურგულაძე გ., **კივილაძე ნ.,** კივილაძე გ. კორპორაციული აპლიკაციების აგება დაპროგრამების სერვის-ორიენტირებული ტექნოლოგიით. სტუ-ს შრ.კრ. „მას“. No 1(21), თბილისი, 2016. გვ.215-220.
8. **კივილაძე ნ.** პროგრამული დანართის არქიტექტურა და RIA-აპლიკაციების დაპროექტება. სტუ-ს შრ.კრ. „მას“. No 1(21), თბილისი, 2016. გვ.243-248.

ABSTRACT

The represented dissertation "Building corporate web-applications using service-oriented architecture", involves the software development process using modern web programming technologies, frameworks and tools. The focus is on the importance of software planning activities and estimating requirements, deciding which architectural style to use and how components communicate with each other. This work provides recommendations, taken from the best practices from the real life scenarios.

The work carries out the discussion about Microsoft's modern web development frameworks and applying them in the development of the large scale line of business applications. ASP.NET Web Forms, Silverlight and MVC programming frameworks are analyzed and provided comparisons between them. The advantages and disadvantages of each framework are carried out and some of the recommendations are provided for using these frameworks in real life scenarios.

Involve discussion about the Microsoft's Business Analysis (BI) platforms and its components, SQL Server Reporting Services tool and functional characteristics. Provide several ways of embedding reporting modules in a web application. A standalone reporting solution is implemented that simplifies and enhances the reporting process. The data is gathered from a variety of sources. The reports are called from a user-friendly, easy to use interface and user is able to export the result in any provided file format: PDF, MS Excel, MS Word. Carry out the ways of adding local reports to the web application or calling server reports using Report Server path. Carry out what factors lead to choose the concrete implementation of reporting integration (users, hardware, architecture and technologies used.

Reveal an importance of well defined software architecture in solving the real life scenarios. Provide software architecture planning stages and some of the general principals of software architecture. Discuss N-Tier/3-Tier design styles. Carry out recommendation of decomposing the design into logical groupings of software components (Layers). Divide an application into separate layers that have distinct roles and functionalities to maximize maintainability of the code and optimize the way that the application works when deployed in different ways.

Provide the key scenarios for using rich client applications, the components found in a rich client application (RIA), and the important design considerations for rich client applications. Provide guidance on the use of appropriate design patterns

(asynchronous callbacks, Command pattern, Service Agent and Proxy) when making design decisions for each category.

Finally, a practical implementation project is provided. A human resource management system (HRMS) web application is a software solution for small to mid-sized businesses to help automate and manage their HR, payroll, management and accounting activities. The application is developed using Microsoft Visual Studio, SQL Server Management Studio and SQL Server Reporting Services software tools. This solution demonstrates the logical grouping of components into separate layers that communicate with each other and with other clients and applications. HRMS web application shows the implementation of the layered design approach. It consists of several logical layers: Data Access Layer (DAL), Business Logic Layer (BLL), Service Layer, Presentation Layer and Cross-Cutting Functionality, that is carried out in the separate layers. Demonstrate the benefits of using Silverlight client to create rich user interface with animations and transformations. Integrate the reporting functionality into the solution, by calling server side reports from the UI and exporting reports in the different file formats. The security is managed through the permissions and roles. Each user has his/her own set of permissions and thus gains an access to each module in a web application. If user does not have a permission for a particular module the access is denied and the module is either invisible or disabled for this user. Also each user action is logged into a special database table and validation checks are performed before each operation.